User Manual for

Client Suite 1.15

© Alle Rechte bleiben vorbehalten, e GmbH

Content

1 Introduction 3
2 Installation 4
2.1 Extract Archive 4
2.2 Copy your Mx' binaries 4
2.3 Configure your Suite 4
2.3.1 UNIX Version – settings.sh 4
2.3.2 Windows Version – settings.cmd 5
2.3.3 GUI behaviour – client.xml 5
3 Simple GUI calls 6
3.1 Client GUI 6
3.2 Remote Client 6
3.3 Admin Monitor 6
3.4 Doc Browser 6
3.5 Service Rights 6
3.6 MxML 7
3.6.1 Monitor Gui 7
3.6.2 Data Dictionary and Templates 7
3.6.3 Validation Queues 7
3.6.4 Error Queues 7
3.6.5 XML Editor 7
3.6.6 Palette Maintenance 7
3.6.7 Workflow Tester 7
3.7 MDCS 9
3.7.1 Cache Monitor 9
4 Some Sample Tasks 9
4.1 Client Macro Examples 9
4.1.1 Record a client macro 9
4.1.2 Reset Password 9
4.1.3 Refresh Settlement Instructions 9
4.1.4 Remove Market Operations 10
4.2 Processing Script Examples 10
4.3 MXML 11
4.3.1 Start & Stop Services 11
4.3.2 Import/Export 11
4.3.3 Flow 12
4.3.4 Palette Maintenance 12
4.3.5 Purge 12
4.3.6 Recovery 12
4.4 Fixings 13
4.4.1 LOAD-PRICES 13
4.4.2 DO-FIXING 13
4.5 Market Data 14
5 Background Information 14
5.1 Script Types 15
5.1.1 XSE - XmlSpacesExchange 15
5.1.2 XRS - XmlRequestScript 15
5.1.3 PSQ - ProcessingScriptQuery 15
5.1.4 XMS - XmlMonitorScript 15
5.1.5 MCS - MxClientScript 15
5.1.6 SPM - Script Parameters 16

© Alle Rechte bleiben vorbehalten, exofin GmbH

1 Introduction

The development of the EXOFIN ClientSuite is driven by two aims:

1. to deliver a versatile Murex Application Client and
2. to deliver examples for scripts and tasks

The EXOFIN ClientSuite provides you with the standard Murex Client functionalities and some new features. Now it is possible to start and stop parts of the Murex environment directly, especially parts of the MxML Exchange module. This allows you to manage your Murex environments a lot more efficient and more comfortable than before.

The ClientSuite contains a batch of example scripts for automation of Murex application and repeated maintenance Tasks, like the MxML purge or EOD processing.
And from now on you only have to configure one file to get a whole set of functions running!
You will read more about the features of the EXOFIN ClientSuite on the next pages!

The EXOFIN ClientSuite exists in two versions, one for Windows and one for UNIX.
The UNIX version is slightly different from the Windows version. The most obvious reason is the difference in the underlying script languages. But also because we assume a different usage of both versions: Windows for ad hoc actions done by real humans and UNIX for automated processing.

Therefore UNIX version has less GUI elements because we assume that you prefer to run your GUI on your local PC rather to run it via X-Window. But you have access on the basic Murex GUI elements like the Session Client or Admin Monitor.
On the other hand the UNIX version is very useful for system integration like calling scripts from a scheduling system during EOD. It has e.g. a result-check for processing scripts with automated notification via e-mail in the UNIX version. (in Windows you can see it in the Debug Modus) included.
Another big advantage is the configuration – please see below.

© Alle Rechte bleiben vorbehalten, exofin GmbH

2 Installation

The installation is very easy. First you extract the downloaded archive, then you put your murex' binaries to the right place and third you configure your client as usual.

2.1 Extract Archive

First you have to extract the files from the archive into a directory of your choice – this directory will be referred to as **<<client-root>>.**
Please assure to use a path without spaces or special characters!

After unpacking the zip-File, there are four folders **_murex**, **\bin**, **\etc** and **\work**:

- The **work**-folder will be used by the Client Suite as the work-directory, where JAR- and DLL-Files will be stored and log files will be written to.
- General Settings will be stored in the **etc**-Folder.
- In the **bin**-folder the program-calls are stored.
- The most important directory for the user is the _murex-folder; all the scripts of the suite are saved in here. This is the main-folder the user works with - all the other folders are actually not interesting for the regular user.

2.2 Copy your Mx' binaries

Copy your mxjboot.jar into the work directory:
<<client-root>>\work.
If you want to use the palette maintenance tool, copy your maintenance.jar into the pm directory:
<<client-root>>\work\pm

2.3 Configure your Suite

All configurations are done in the etc-Folder.
So this is by default the only place where you need to amend something. In addition a task under **_murex** can be adapted to your needs. All other folders/files shall not be touched.

Do not modify anything else as in <<client-root>>\etc and <<client-root>>_murex!

2.3.1 UNIX Version – settings.sh

A big advantage you have on UNIX side is that you can directly access the Murex configuration files. That enables the EXOFIN ClientSuite UNIX to configure itself automatically! ☺
You have only to define the absolute path of your ClientSuite and of the Murex Home directory in the **etc/settings.sh** file of your ClientSuite installation. The rest will be done by the ClientSuite itself!
You may have to adopt the MXJ_PROCESS_NICKNAME and MXJ_PLATFORM_NAME if you want to use the GUIs and in case you have defined your own nick name different to the defaults as in launchmxj.app.
For more details see the comments in the **settings.sh**.

© Alle Rechte bleiben vorbehalten, exofin GmbH

2.3.2 Windows Version – settings.cmd

You have to enter your environment settings in **etc\settings.cmd**.
All variables are marked like this: "__xxx__", so you can easily search for them by using the pattern of two underscores "__".

In case you are not using parts of Mx e.g. Documentation Server or Excel Bridge, you can leave these fields as they are.

In order to set a general log level or to start in debug mode, you can change values in this file.
For starting single scripts with log level or debug mode you can change the respective value in the scripts under <<client-root>>**_murex**.

2.3.3 GUI behaviour – client.xml

In the file **etc\client.xml** you can customize your individual Murex session. The file is the same for both UNIX and Windows version. It contains a good annotation of all the features that we are aware of and so the file should explain itself. Additional comments and samples will help you to get the idea of each entry.
Here we will shortly comment the basic features that you need to adjust your GUI in the client.xml file.

2.3.3.1 GUI window - Screen

The window that appears by starting the default Murex Client can be configured via the node **Screen**[1].
Here you can set the following sub nodes:

Font
With the Font-Node you can configure the font type that is used in your Murex Client. The default setting of the Client Suite is "Tw Cen MT" – so if you don't have this font installed on your PC you should change it.

GUI Window size
The Dimension-Node gives you control over the screen size of you Murex client. You can configure the width and the height. The default value of the EXOFIN Client Suite is 900 x 700 pixels.

Session title and Session info
The Title-Node lets you configure if in the GUI window additional information about your individual Murex session is displayed or not. This text is shown in the window title and also in your windows task bar. So you can directly see if you are on a production or a test environment, good to know sometimes! ☺
Setting **AddMxInfo** to "Y" the GUI will give you information about the session ID. This can be very helpful if you work with remote control of Murex sessions[2].
Setting **AddUserInfo** to "Y" information about your current user group and user is shown.
The default values are "Y" for both attributes and the Session Title is: PRODUCTION-SUPPORT

2.3.3.2 Additional Config

Just to mention one other thing that came up as an important point in our experience is the configuration of the locale.

---

[1] /ClientConfig/Screen
[2] sf. 3.2, Remote Client

© Alle Rechte bleiben vorbehalten, exofin GmbH

Depending on the **locale settings**[3] you choose in the **client.xml** the decimal display of numbers varies. For example if you chose "DE" for German, numbers will be separated by "," as decimal delimiter and "." as thousands separator. If you chose "EN" the English convention will be displayed, just the other way round with "," as thousands separator and "." as decimal delimiter.
The default value of the Client Suite is "DE", so be aware of this.

3 Simple GUI calls

Of course everything starts by simply launching the Murex Client (**Client_GUI.cmd**).
But there are other GUIs that you'll might get used to because they are comfortable shortcuts to modules of Murex' huge client.

3.1 Client GUI

OK, as we said: It all starts with the standard murex client: **_murex\Client_GUI.cmd**

3.2 Remote Client

And already the next feature is one that most people are not aware of: You can control a GUI remotely! Murex gives the opportunity for a user to provide access to his session to a remote user. This can be especially helpful if he needs help from a remote support desk or if he would like to launch a heavy procedure and be able to monitor it remotely. (e.g. : support tasks during night from home office) .
Of course the user of the session to be controlled has to grant access first via "UI Tools" → "Monitor" → "Grant Remote Access Right".
Then he gives the session ID (SID) to the user that wants to control the session. The given SID has to be entered after launching the **_murex/Client_GUI_to_remote_session.cmd**.

Annotation:
The SID is normally shown in the window title if set so in your etc\client.xml configuration (see client.xml):

```
<Title AddMxInfo="Y" AddUserInfo="Y"/>
<!-- AddMxInfo shows PID/NPID/SID, AddUserInfo shows group and User, all will be shown with title in header -->
```

3.3 Admin Monitor

This is the well known admin monitor (called monit.cmd in the murex standard distribution) that allows you to monitor the different MxML-Services. Start it from here: **_murex\Admin_Monitor.cmd**

3.4 Doc Browser

And most likely you know this already, too. It's the client of the documentation browser you can start also from within the Client by pressing F1 – or as separate process without a session by using **_murex\Doc_Browser.cmd**

3.5 Service Rights

This script leads directly to the GUI where you can define templates for the rights for services such as the MxML Dictionary. These templates can later be assigned to certain groups via a supervisor session.

[3] /ClientConfig/Locale

© Alle Rechte bleiben vorbehalten, exofin GmbH

3.6 MxML

In order to run the following MxML-GUIs you need to specify the user-group you want to use for the task in the appropriate script or in the general config file (etc/settings.cmd). The Parameter is MXJ_GROUP_CODE.

3.6.1 Monitor Gui

This opens the mxml monitor with full access to all areas as you know it from the Client GUI by choosing menu Processing → Document Workflow → Monitor.
If you are using this Murex feature intensively you will love this script for the unnecessary login and faster GUI load. ☺
Try it: **_murex\mxml\Monitor_GUI.cmd**

3.6.2 Data Dictionary and Templates

These scripts will open just the DataDictionary or the Templates of the MxML-Exchange workspace in a separate window.
Data Dictionary: **_murex\mxml\Data_Dictionary.cmd**
Templates: **_murex\mxml\Templates.cmd**

3.6.3 Validation Queues

You have also the possibility to start the Valdiation Queues directly, so without the GUI. It will automatically start with the group that is assigned to you.
Here you go: **_murex\mxml\Doc_Validator.cmd**

3.6.4 Error Queues

This script opens the Error Queue tab which contains two tabs for each data source entity: "Processing Errors" and "Lost Data". This feature is very practical to look up and purge errors without open the MxML Exchange Monitor.
Here you go: **_murex\mxml\ Doc_ErrorQueues.cmd**

3.6.5 XML Editor

Murex has its own XML Editor. With this script you are able to use it also without starting Murex. If you are working on a system without any XML Editor, it could be very usefully to use the one from Murex.
Here you go: **_murex\mxml\XmlEditor.cmd**

3.6.6 Palette Maintenance

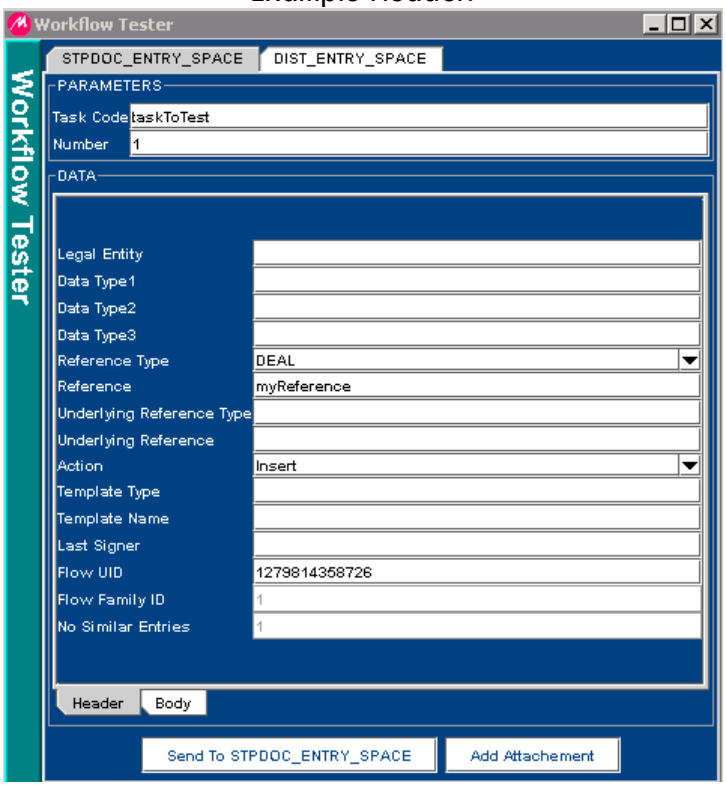
With the PaletteGUI.cmd in the palette directory you can start the palette maintenance tool for importing new task types or merging during upgrades. See also Palette Maintenance in the tasks section. To start the GUI start this one: **_murex\mxml\palette\Palette_GUI.cmd**

3.6.7 Workflow Tester

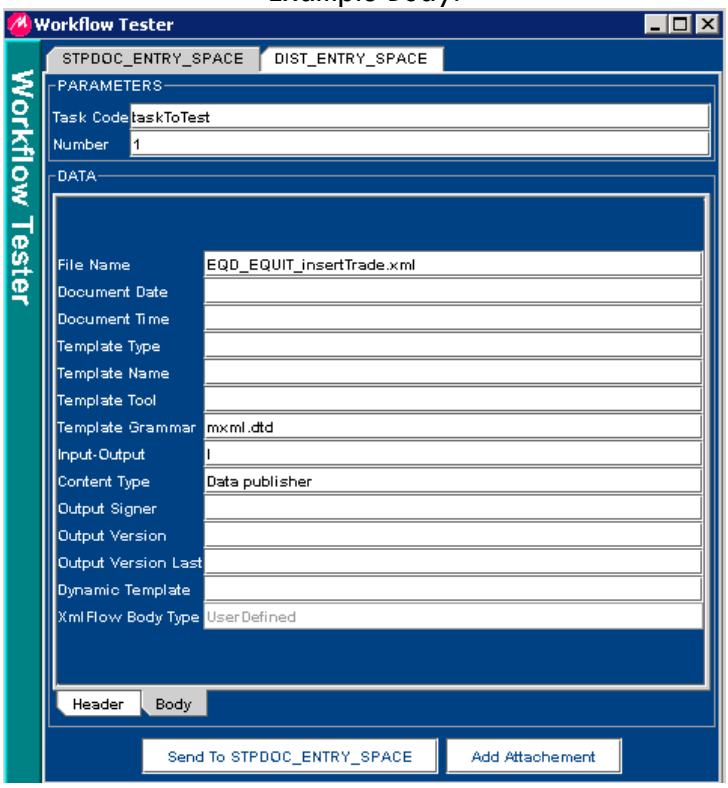
With the **_murex\mxml\WorkflowTester.cmd** you can access a useful Murex tool for testing purposes of individual Workflow Tasks. You can easily send specified deal tickets to a single task of your workflow. You only have to store a deal ticket file in the ClientSuite **work** directory, then define the file name and some common fields and click on "Send to STPDCO_ENTRY_SPACE" – that's it!

© Alle Rechte bleiben vorbehalten, exofin GmbH

Example Header:



Example Body:



© Alle Rechte bleiben vorbehalten, exofin GmbH

3.7 MDCS

3.7.1 Cache Monitor

With **_murex\Cache Monitor.cmd** you can watch the cache of Mx that is defined by the service MXCONTRIBUTION.CACHE. The Cache Monitor is a useful tool when you are using the MDCS interface to upload historical market data into Murex, as you do normally during the Fixings procedures. In the Cache Monitor you also will find your XML tickets from updating Market Data via MDCS.

4 Some Sample Tasks

In this chapter we give a short overview of the features. In certain cases you can do adaptation. These possible adaptations will also be illustrated.

4.1 Client Macro Examples

The following examples show you different tasks that can be done by Murex macros. They also give you a short introduction to MxML Scripting on which Murex macros are based. We start with a simple macro and go on to some more comprehensive ones. So you should be able to create your own macros in a more systematic way.

4.1.1 Record a client macro

Generally you can record your own client macro with the script **_murex\util\clientMacro_record.cmd**. When you start it, you will be asked where to store the result file. If you enter nothing a file macro.xml will be created in the working directory **work\macro.**
This is always the basis for further scripting.

4.1.2 Reset Password

This is the first example of a Mx client script. Here you see that you can separate configuration from the script itself by using an additional file. You can find it under **reset_pw**. It just uses script parameters for the configuration inside the file **config.spm.xml**.
Put in the supervisor password (encrypted[4]!), then the name and password of the user to be changed and run the script – that's all!

4.1.3 Refresh Settlement Instructions

This is an example for a recorded macro that refreshes settlement instructions of a specified counterparty. It gives you an impression on how to use the tag list elements (as they are called) and how you can access them in your macro script. You can find it under **_murex\refresh_SSI**.
Just adapt the configuration of the macro client script – **config.mcs.xml** – and put in the label of your counterparty.
As in the example above you have to specify the parameters you want to run the script with – **config.spm.xml** – and you're ready to refresh SSI without the need of logging into the Murex GUI.
Of course you could extend the scripts so that you refresh SSI for all counterparties during the End of Day. ☺

[4] You can find a Passwort Generator the Admin Monitor

© Alle Rechte bleiben vorbehalten, exofin GmbH

Important:
Please notice that this script is only an example that was included into the Exofin ClientSuite to show you what can be done by Murex scripts. Take care to configure the script properly and test it on your Murex environment for testing purposes before using it in production!

4.1.4 Remove Market Operations

This is another example of a Mx client script. You can find it under **_murex\remove_MOP**. This example is a more advanced one. It will show you how you can use a list of elements e.g. deals and doing the same operation on all of them using the loop functionality of Murex macro.
The only thing you have to do are the following four steps:

1. As usual configure the user details in config.spm.xml (User, Password, User-Group)
2. Define the trades that should be considered in the config.mcs.xml file (in our example trades where you want to remove a XIT operation)
3. Adapt the LoopsCounter in the script.mcs.xml and adjust it to the same value as how many trades you've put in the config.mcs.xml.
4. Run the script run.cmd

4.2 Processing Script Examples

This chapter gives you a first impression what can be done with Murex processing scripts, mainly used for the EoD procedure. You will find our examples for processing scripts in the directory **_murex\eod** in your EXOFIN Client Suite.
Usually the EoD will be run by a scheduling tool and most likely on the server side, but sometimes you like to re-run something (e.g. to roll the Front Office Date) and with the client suite it is quite easy. If you run processing scripts with the client suite, you get a message in your console (Windows Version) or e-mail notification (UNIX) whether it ran successfully or not.

Furthermore, the script compilation can be a good starting point to add your own EoD tasks and write a meta-script that runs them all at once in the proper order – then you can run your whole Murex-EoD directly from your client.

The script compilation contains four different examples that are calling a processing script. The **FODESK-EOD** and **PROCESSING-EOD** scripts just refer to the standard Murex' Processing Scripts that move the dates, i.e. Front-Office-Date and Processing-Center-Date.
In addition we put in a script to run the fixing procedure, you will find the script in **_/murex/fixing/FIXINGS-EOD** directory. This script can be used in conjunction with the corresponding LOAD-FIXINGS script – please see Load-Fixings, which enables you to upload historical market values.
The fourth example is described in Recovery.

The configuration of the processing scripts which should be executed by the client scripts is done in the corresponding **config.psq.xml** and **script.xrs.xml** files. There you have to declare it by its label that you had assigned to it in Murex during the processing script set up. In the **script.xrs.xml** you have to adapt the service nick name under which the processing scripts are running.

Note:
If you use the ClientSuite Unix version, you can automatically perform checks on the created log file and use the implemented notification feature ☺. For more details see XRS - XmlRequestScript, please.

© Alle Rechte bleiben vorbehalten, exofin GmbH

4.3 MXML

4.3.1 Start & Stop Services

With these scripts you can start defined services of the Murex environment.
We made two subfolders for your convenience:

1. One to start/stop the STATICSREPOSITORY: **_murex\mxml\Services\STATICSREPOSITORY**
2. One to start/stop all services: **_murex\mxml\Services\All**

In the **start.xms.xml** and **stop.xms.xml** you do the configuration. You have to specify your Monitor-User and his password:

```
<MonitorUser>
      <Name>ADMIN</Name>
      <Password>___your decrypted password___</Password>
</MonitorUser>
```

And you can define a block for each services to be started/stopped:

```
<MonitorCommand Type="START_SERVICES | KILLALL">
      <Service>
            <PlatformName>MX</PlatformName>
            <NickName>MXDICTIONARY.FINANCIALPARSER</NickName>
            <EndManagement>
                  <ErrorFile/>
            </EndManagement>
      </Service>
</MonitorCommand>
```

4.3.2 Import/Export

Here you can import/export all objects of MxML Exchange.

After starting the script, the user has to enter the path and the name of the export-/import-file.
You can either

- enter a complete path and filename (then it will be saved there) or
- enter a path relative to the working directory or
- just a name (this will store the file in the working dir) or
- nothing (then it will be saved with the shown default path and name).

By default all files are expected in the mxml-subdirectory of the working dir.

This way you can easily make a full copy of MxML from one environment to another. Just double-click all object tasks once for export on the source environment and once for import on the target environment. Or if you have this need more often: Just write a small script that does all the calls! ☺

© Alle Rechte bleiben vorbehalten, exofin GmbH

4.3.3 Flow

In the folder **_murex\mxml\flow** you can find some examples for processing scripts that allow you to start and stop the

- whole flow (subfolder **All_Tasks**)
- single (subfolder **EVS**) or
- several tasks (subfolder **FileImport**)

without starting any GUI.

4.3.4 Palette Maintenance

With this tool you can start the palette maintenance tool for creating new task types or merging during upgrades.
You have two options here: First is to start the GUI (see Palette Maintenance in the GUI section) and second is to specify an action file and the palette.jar in advance and perform your actions directly without GUI (helpful if you have to update several environments).

4.3.5 Purge

4.3.5.1 readInfo

This script reads the information of the MxML Purge Monitor and represents it as a webpage. This way you don't have to login into the admin monitor, wait for the GUI dialogs to be built/refreshed and you have a document that you can provide or simply use it for documentation of your purge.

4.3.5.2 doPurge

doPurge performs a MxML purge and gives you the information about the status before purging and afterwards in an html file. It will also show you the date of the purge and how many tickets are affected by the purge. This gives you easily the information you need from your purging operation!
You only have to specify what and until which date to purge in the config.psq.xml.

4.3.6 Recovery

If there is a failure in MxML Service, tickets are not sending to MxML Exchange and stored in the Recovery Monitor. The action to resend these tickets to MxML Exchange can be done automatically by using a processing script in **_murex\eod\RECOVERY**. Murex is using the unit "MXML Recovery" for this script. In the **config.psq.xml** you can choose the name of the corresponding Mx processing script you have assigned before in MX:

```
<ProcessingScriptQuery>
      <Script>
            <Name>RECOVERY_Tickets</Name>
            <Predefined>Yes</Predefined>
      </Script>
</ProcessingScriptQuery>
```

In our example the name of the processing script is **RECOVERY_Tickets**
For more details how to configure Murex processing scripts refer to Processing Script Examples.

© Alle Rechte bleiben vorbehalten, exofin GmbH

4.4 Fixings

The EXOFIN ClientSuite supports you by doing the fixing procedure. It offers you an easy way to upload the historical market values and enables you to run the fixing procedure by a script. The next chapter shows you how to do it, starting with the market value upload.

4.4.1 LOAD-PRICES

Via **_murex\fixing\LOAD-PRICES\run.cmd** you can load historical data over MDCS interface into Mx. Afterwards you can run the fixing procedure via the client script also – see below.

You find sample files in **_murex\fixing\LOAD-PRICES\Samples** including historical data which can be imported. This files are only examples so you have to adapt it according to your environment. You can also use it as templates for testing the MDCS interface. In the run.cmd you have to paste the correct filename:

```
SET IMPORTFILE=%DIR_TASKS%\fixing\LOAD-PRICES\Samples\#Sample File#
```

In order to test the templates you have to adapt the name of the processing script in the config.psq.xmlfile in _murex\fixing\LOAD-PRICES

```
<Script>
            <!--defines the used processing script-->
            <Name>#processing script#</Name>
            <Predefined>Yes</Predefined>
</Script>
```

4.4.2 DO-FIXING

The script enables you to perform the Murex fixing procedure by using a processing script. To start the processing script you only have to click on the **run.cmd** in the **_murex\fixing\DO-FIXING** directory.
In the **config.psq.xml** you can choose the name of the Mx processing script, that imports historical data (that you may have loaded before via Load-Fixings ☺) from MXCONTRIBUTION.CACHE into Mx:

```
<ProcessingScriptQuery>
      <Script>
            <Name>Import_FX_Spot</Name>
            <Predefined>Yes</Predefined>
      </Script>
</ProcessingScriptQuery>
```

In our example the name of the processing script is **Import_FX_Spot**.
For more details how to configure Murex processing scripts refer to Processing Script Examples.

© Alle Rechte bleiben vorbehalten, exofin GmbH

4.5 Market Data

In this chapter you will find support for the import of market data into Murex. This offers you an easy way to sent XML tickets to the MDC Server and update market prices plus other types of market parameter.

Via **_murex\marketdata\run.cmd** you can send a XML ticket to the MDCS interface. The script also starts a processing script to import the data into murex.

You find templates for XML tickets in **_murex\marketdata\Samples** including examples for different asset classes and data types. Use subtrees from the templates and adapt it according to the market data defined in your environment. Save them as XML files in **_murex\marketdata\Samples** and paste the correct filename in the run.cmd:

```
SET IMPORTFILE=%DIR_TASKS%\marketdata\Samples\#Sample XML file#
```

Due to the flag MXJ_ADD_IF_NOT_EXIST in bin\mdcs\importCache.cmd is set to "Y", it is not necessary to export market data to the MDC Server before you import some. Anyway you have to configure your Provider in MX.

In order to test your tickets you have to adapt the name of the processing script in the config.psq.xml file in _murex\marketdata.

```
<Script>
            <!--defines the used processing script-->
            <Name>#processing script#</Name>
            <Predefined>Yes</Predefined>
</Script>
```

For more details how to configure Murex processing scripts refer to Processing Script Examples.

5 Background Information

In the **_murex**-folder of the Windows-Version are all files executable of type .cmd and can be started by simply double-clicking the icon.
For some scripts the needed xml-files are in the same folder too.
These XML-files are all sample-files, so you have to configure those files manually to fit with your needs.
A three letter suffix before the .xml-ending indicates the type of XML-File that is used in order to help understanding the files. The different types are explained below.

© Alle Rechte bleiben vorbehalten, exofin GmbH

5.1 Script Types

5.1.1 XSE - XmlSpacesExchange

This type is used to define the export of MxML Spaces Objects.

5.1.2 XRS - XmlRequestScript

This is used to tell the processing scripts, what they have to do, where to find the appropriate configuration file and where to write the logs to. Please see PSQ, too.

XRS – Notification:

Until now this feature is only available in the UNIX version. The notification is related to the Murex Processing Scripts. But what is this feature? During the execution of a Murex Processing Script a log file is created with information about success or failure of your Processing Script. The notification feature checks the created log files and send out an email to a set of recipients using UNIX based mailx function. You can also define which mailadress should be displayed as senderadress.
You can configure the notification mode in the **settings.sh** under the **etc/** directory of your EXOFIN ClientSuite installation. See this file for more information.

5.1.3 PSQ - ProcessingScriptQuery

These are the configuration files for processing scripts that perform the effective tasks.

With the suite you will get some processing scripts as examples that will e.g. start or stop the complete flow or purge the MxML spaces.
The configuration of the script itself is done in the xrs.xml-files (XmlRequestScript). The configuration of the effective task is done in the psq.xml-files (ProcessingScriptQuery). Here you specify what the task shall do, e.g. what services to be started/stopped, or what settings to use for a purge.
Here you can also define in the Unix version the SQL trace log level and the name prefix of the trace log file.

5.1.4 XMS - XmlMonitorScript

Used to define services to start and stop.

5.1.5 MCS - MxClientScript

This is a script where macros are recorded to and that can be played back afterwards[5].

For this type of scripts we included some examples - one resets the password of a user, the other refreshes settlement instructions between two parties.
In the refreshSSI-macro there two ways of configuring the macro-script are shown. You can either use TagListElements to define fields (config.mcs.xml)or ScriptParameters (config.spm.xml). As you can see in this script, both files can be used together.
The TagListElements you can access during a loop for example, so that in every loop a different element is chosen.

Please also see Mx Documentation for further details on macros!

[5] Also used for configuration of some macro-details

© Alle Rechte bleiben vorbehalten, exofin GmbH

5.1.6 SPM - Script Parameters

In these files the parameters for the mx client scripts are be stored as explained above. Also refer to the sample tasks provided with this suite!

© Alle Rechte bleiben vorbehalten, exofin GmbH